CHEETAH DIGITAL

Cheetah Loyalty HTML Games

Developer Guide

Version	Date	Description	Reviewed / Approved by
1.0	August 2019	Initial release	Cheetah Digital Product Management
1.1	September 2021	Added version history	Cheetah Digital Product Management

Contents

Introduction

This document describes how to build interactive HTML games and deploy them on Cheetah Loyalty.

Audience

The intended audience is marketers and game developers. Some of the setup steps also require technical knowledge such as JavaScript, HTML, CSS.

Feature Highlights

Cheetah Loyalty provides powerful capabilities for building rich interactive games to reward and please customers.

- **Cheetah Loyalty Game Framework (aka Stellar Game JS):** simple, flexible JavaScript interface between the game and the web or mobile app container
- **Web and Mobile:** games can be played on both web and mobile apps because they are written in HTML 5.
- **Sample Games:** included sample games are spinning wheel, scratchcard, shell game, darts, and hidden object, with more added in every Cheetah Loyalty release.
- **Game Frameworks:** Sample games are built on Phaser.js, an HTML 5 game framework, but any JavaScript-based game could work.
- **Flexible Hosting:** games can be hosted by Cheetah Loyalty or as an external URL
- **Cheetah Loyalty Game Challenges:** each game is launched by a Cheetah Loyalty Game Challenge, which provides flexible configuration options for styling, behaviors, and prizes
- **Prize Configuration:** multiple prizes can be configured per challenge, and awarded based on random or probability configuration options

Part 1: Game Development Process

Follow this process to build and deploy a game. The tail-end of the process involves formal involvement from your Marketing Operations team in order to assure a consistent and reliable experience for your members.

This process is simpler if you are customizing a sample game provided by Cheetah Loyalty instead of writing your own, because you only need to configure the game and produce image assets. So for your first game you should customize a sample game.

Roles & Responsibilities

- **Marketing Lead:** your marketing/business owner responsible for defining the game and obtaining various approvals from the customer
- **Game Developer:** your Game Developer is responsible for implementing the game, producing all assets such as HTML, CSS, JavaScript, images, sounds. The Game Developer produces a release-ready "Candidate" game on the Staging Env that is fully complete, polished, and tested.
- **Marketing Operations:** your Marketing Operations team is responsible for providing support, testing, production rollout and ongoing support. Most of this is done in the Cheetah Loyalty Marketing Console, web and mobile apps.

Phases

#	Phase	Responsible	Description
1	**Specification**	Marketing Lead	Written **Game Specification** document for Marketing Operations to review.
2	**Prototype**	Game Developer	Live working example game on Staging Env for early internal reviews/iterations.
3	**Configuration**	Game Developer	Game Developer configures game on Staging Env to match Game Spec, including final art, copy, and prizes, exactly as what will go to production. Game Developer unit tests game and declares when the game is ready for final testing.
4	**Testing**	Marketing Operations	Final QA/testing in Staging Env.
5	**Approval**	Marketing Lead	Approval from customer to release game, possibly following a UAT.
6	**Release**	Marketing Operations	Config and release in Production Env.

Part 2: Game Specification

The Game Specification is a written description of the game, containing important high-level concepts such as business goals, marketing concept, possible types of game, and prizes.

Putting your ideas into writing is crucial to the success of your game:

- Your customer can provide feedback and approval
- Your internal stakeholders can understand your game, provide feedback
- Your marketing team can better coordinate their activities to support game launch
- Your marketing operations team can review your game spec and guide you on best practices
- Your marketing operations team can test your game
- Your marketing operations team is best positioned to launch and support your game

Use the following template or similar to write your Game Specification.

Game Specification Template

<u>Game Concept and Name</u>

What is the game concept and working name of your game?

<u>Business Goals</u>

How will this game help your loyalty program?

<u>Milestones/Dates</u>

1. Game Spec Date: (at least 3 weeks before Release)
2. UAT Date: (allow 2+ weeks after Game Spec)
3. Release Date: (allow 1 week after UAT)

Please plan to allow 2+ weeks for UAT after submission of the Game Spec, and then 1+ week before Release. Allow additional time for custom game development or custom creative, see below.

<u>Team</u>

- Marketing Lead:
- Game Developer:
- Marketing Operations:

<u>Audience</u>

Which members can play the game?

- Segments / Audience rules:
- Limits:
- Code submission required? (please describe)

<u>Game Type</u>

What kind of game will this be? Select one of the options below.

- ❑ Cheetah Loyalty Game
  - ❑ Spinning Image Wheel Game
  - ❑ Scratchcard
  - ❑ Shell Game
  - ❑ Find the Hidden Object
  - ❑ Darts
  - ❑ Star Trek Transporter
  - ❑ Plinko
  - ❑ Shake it Up
  - ❑ Other Existing Game (which one?)
  - ❑ New Cheetah Loyalty Game (describe, and plan to allow 2+ additional weeks)
- ❑ External Game
  - ❑ Existing Game (which one?)
  - ❑ New External Game (describe, and plan to allow 3+ additional weeks)

Game Skin

Describe the game skin.

- ❏ Using default/existing game skin
- ❏ External skin (I will provide)
- ❏ New or custom Stelar skin (describe, and plan to allow 1+ additional weeks)

Game Content

- Heading:
- Subheading:
- Body:
- Details:
- Image:

Prizes

List all prizes and prize probabilities in the table below. Common prize awards are metrics (points) or contest (rewards).

Prize Display Name	**Prize Award**	**Chance of Winning [1]**	**Limits / Details [2]**
2 points	2 points	70%	
3 points	3 points	15%	
5 points	5 points	10%	
10 points, yay!	10 points	5%	
No Prize	--	--	

- [1] Chance of Winning percentages should add up to 100%
- [2] List the prize Internal Name if you are building a custom game.

Other Requirements

List any new requirements or open issues/questions.

Part 3: Cheetah Loyalty Game Interface

Your game must include **stellar-game.js** and implement the interface methods and callback methods defined below. This provides bi-directional communication between the web or mobile container app and the game.

This interface will remain stable between versions of Cheetah Loyalty, although the specific implementation may change. See Appendix B: stellar-game.js for the current version source code as well as the sample implementation code. The source code for all sample games is also available in your program hosting and reviewing their implementations should be very informative.

Interface Methods

You need to implement the interface methods and register them with Stellar.game.

Initialize Prize Handler

This function will be called by the game container when your game index.html page loads. Copy/paste the following code into your game and implement the function.

```
// copy/paste this function into your game and implement
Stellar.game.onInitializePrizeHandler = function(prizes,gameConfig){}
```

The parameter gameConfig represents various configuration options for the game runtime. This comes from the Challenge Definition JSON (see Part 3).

💡 **Tip:** Try to avoid hardcoding any potentially useful configuration options into your game, especially images such as foreground, background, etc. Write your game so that these options are obtained from the gameConfig parameter. See Appendix A: Game Configuration for some tips and strategies.

Submit Challenge

Call this function to award the prize. This submits the challenge and the member receives a prize as configured in the challenge. This is an asynchronous function; the Finish Challenge Submission Callbacks (below) will be used when the challenge submission finishes.

```
// copy/paste this function into your game and call when appropriate
Stellar.game.submitChallenge();
```

Finish Challenge Submission Callbacks

These are success/failure callback functions that will be called by the game container when Stellar.game.submitChallenge completes.

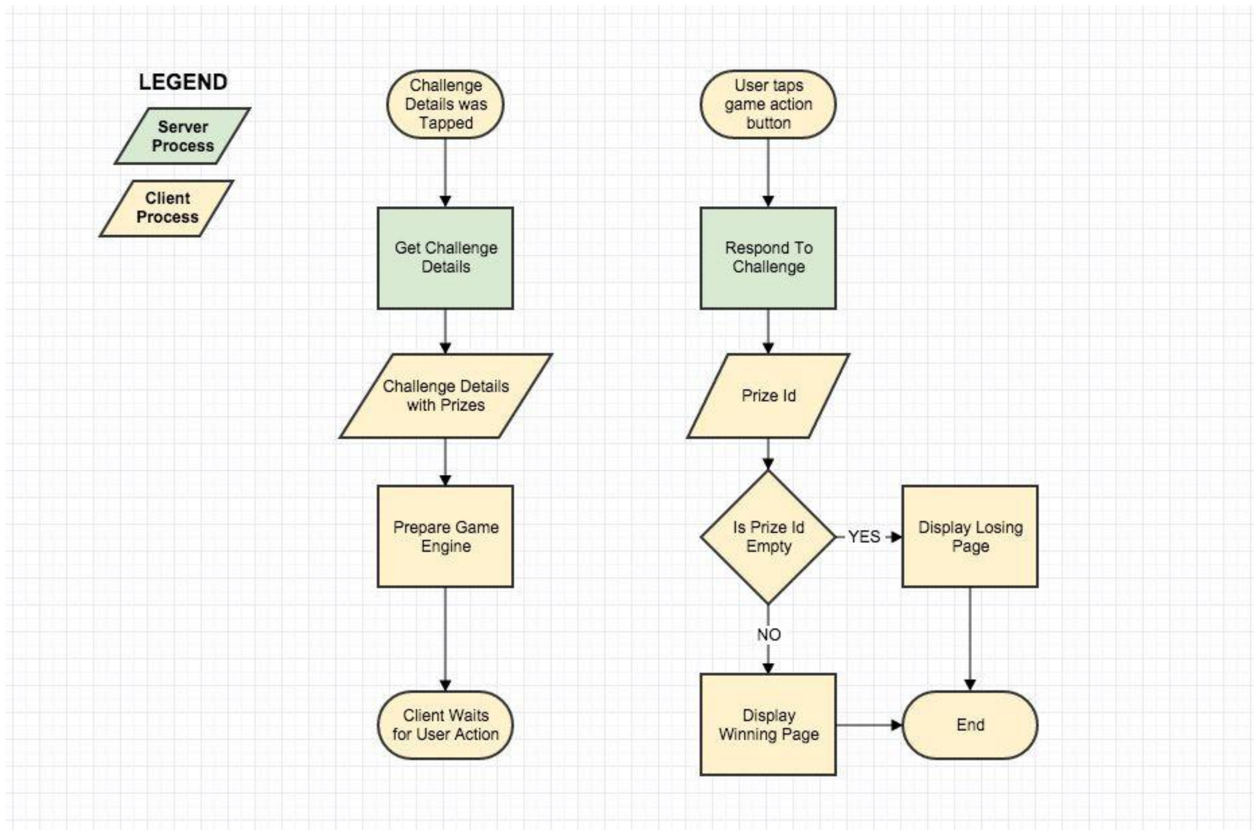
```
// copy/paste this function into your game and implement it
Stellar.game.onFinishChallengeSubmissionHandler = function(prizes,prizeId){
game.showResult() }

// don't override the default implementation of this function unless you want
to customize error handling
Stellar.game.onFinishChallengeSubmissionWithErrorHandler = function (error){
game.showResult() }
```

showResult

Call this function when the game is done and you want to show the prize. The stub implementations for the challenge submission callbacks (above) simply call this method immediately, but your game should probably override `Stellar.game.onFinishChallengeSubmissionHandler`.

```
// copy/paste this function into your game and call when appropriate
Stellar.game.showResult();
```



Part 4: Cheetah Loyalty Configuration

This section describes how to configure games in your Cheetah Loyalty environment.

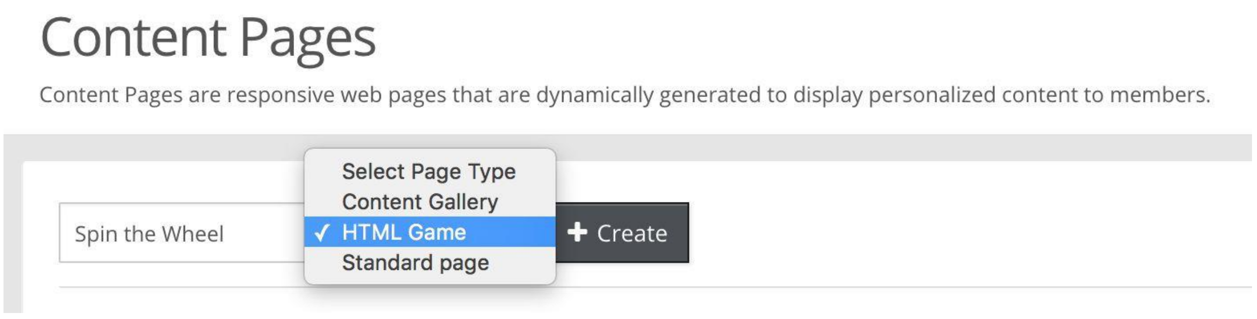
Each game is configured as a Content Page, which can then be used by one or more Challenges.

Step 1: Content Page

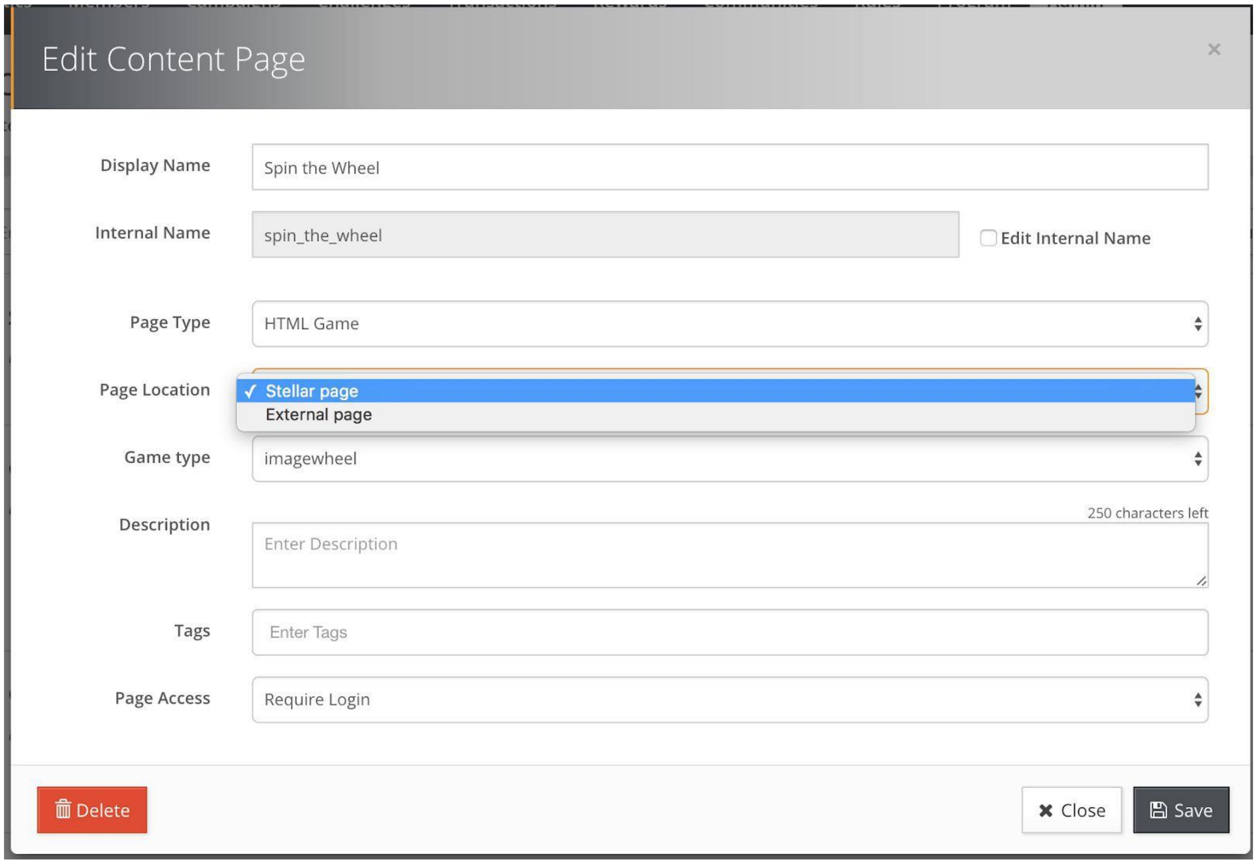
First create a game content page. You only need to do this once per type of game.

⚠ Warning: Development should only be done in your Staging Env, not your Production Env. Refer to Part 1: Game Development Process; you are probably in Phase 2.

1. Go to Admin > Content > Content Pages
2. Add a new Content Page of type "HTML Game"
3. Choose a hosting option of either a Cheetah Loyalty Page or External URL
4. For Cheetah Loyalty Page hosting, upload your game assets as Static Files. This must include an index.html page that loads the stellar-game.js framework. Your index.html file can load additional resources such as css or js, so long as the paths are absolute.
5. For External hosting, enter the URL of the game index page. Please note that for browser security purposes, your game must be hosted in the same domain as the web app. (You can work around this in development by modifying your hosts file.)


Content Pages

Content Pages are responsive web pages that are dynamically generated to display personalized content to members.



Display Name	Spin the Wheel
Internal Name	spin_the_wheel ☐ Edit Internal Name
Page Type	HTML Game
Page Location	✓ Stellar page / External page
Game type	imagewheel
Description	Enter Description (250 characters left)
Tags	Enter Tags
Page Access	Require Login

Step 2: Challenge(s)

The "games" visible to members in web and mobile apps are actually Cheetah Loyalty Challenges. Web and mobile apps usually show a list of challenges, although not all of them will be games and the member-visible name of the list or page might be something like "Earn" instead of "Challenges".

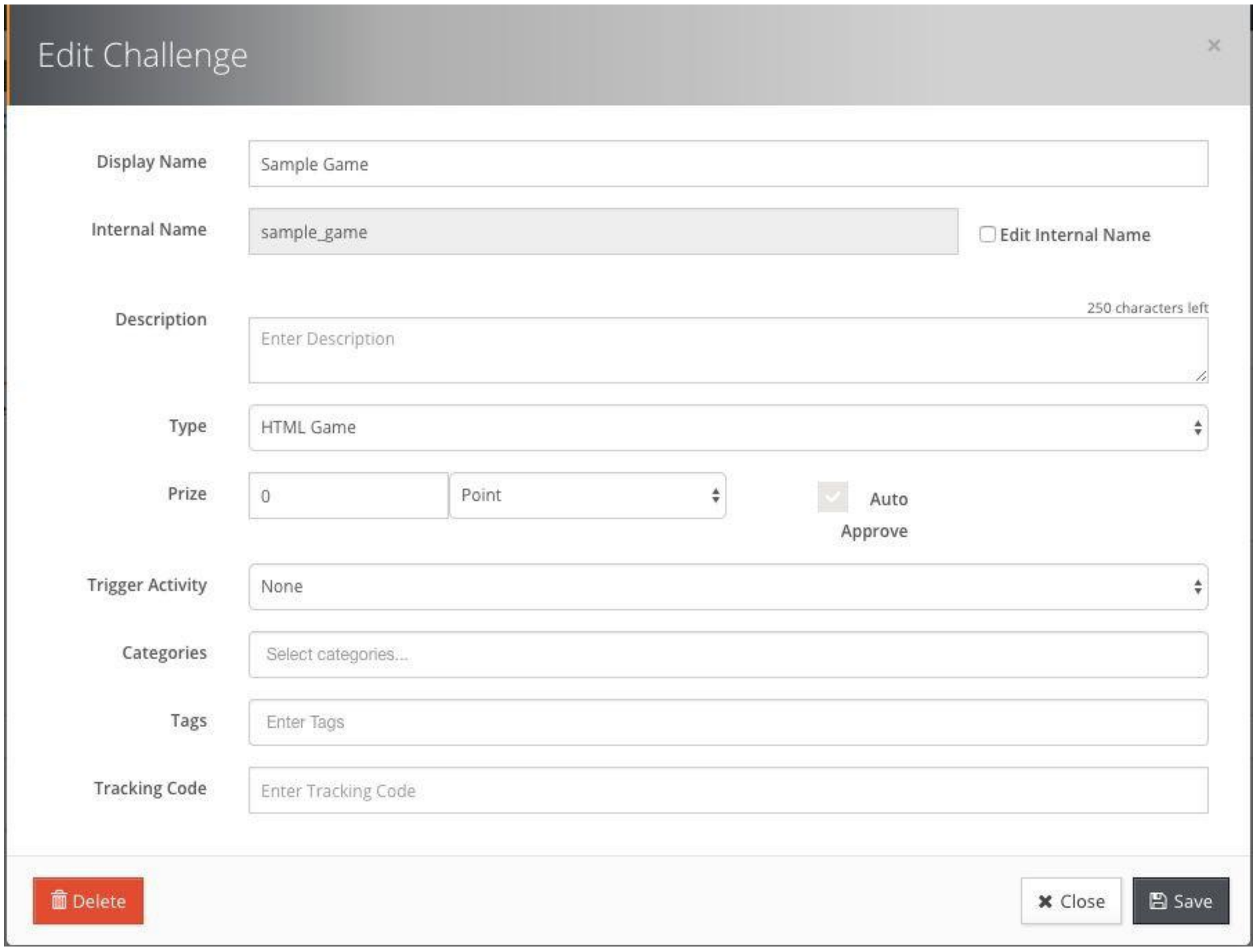
The challenge contains the marketing copy/graphics visible in the list, as well as configuration options and prizes for the game. This allows a single content page game to be flexibly used in multiple challenges.

⚠ Warning: Development should only be done in your Staging Env, not your Production Env. Refer to Part 1: Game Development Process, you are probably in Phase 2.

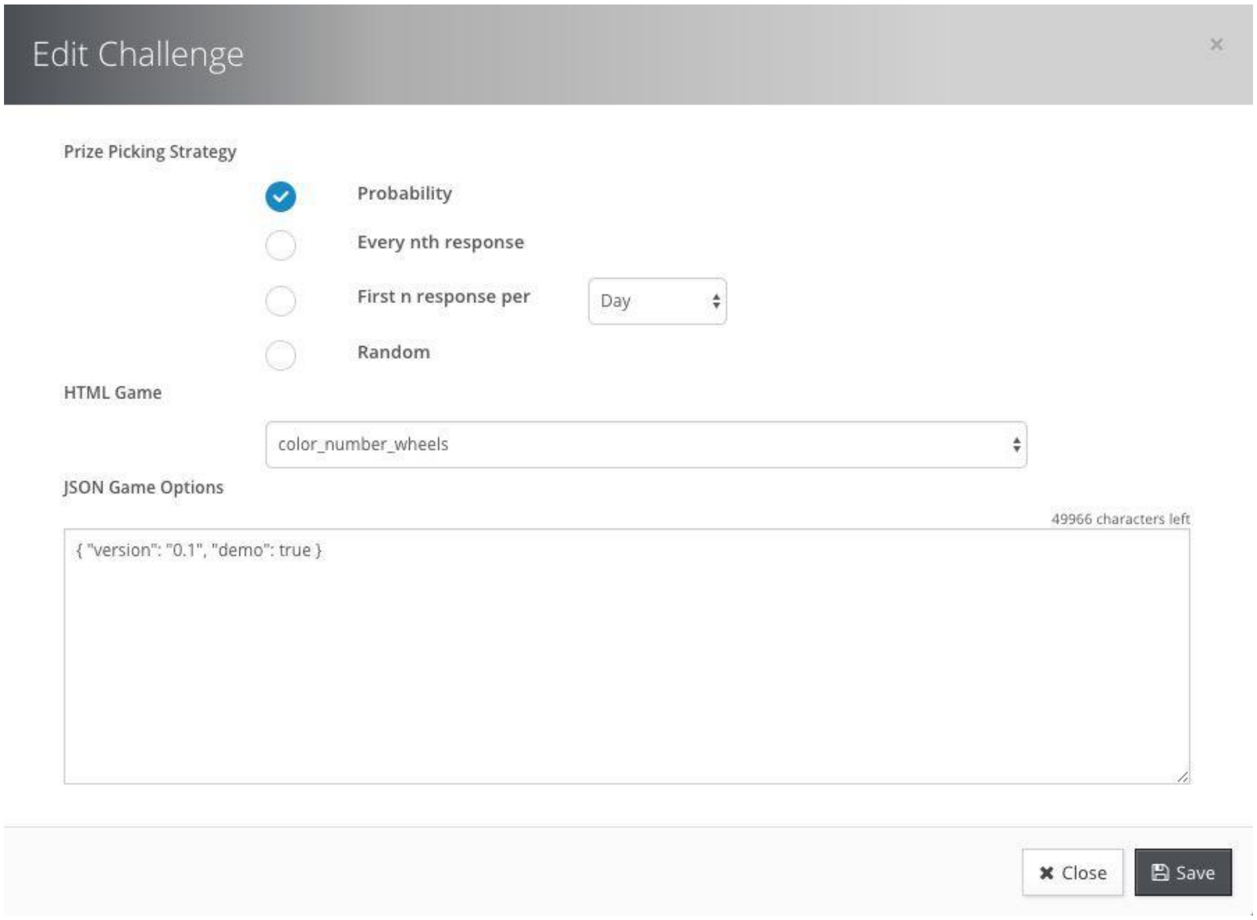
1. Go to Admin > Challenges
2. Add a new Challenge; set **Type = HTML Game** and click Save
3. Open the challenge Definition
    a. Set HTML Game to your game content page

   b. Set the Game JSON to a stub for now, such as
      `{ "version": "0.1", "demo": true }`
   c. Click Save
4. Configure at least one Prize. A simple prize of “100 points” is good to start with.
5. Open the Display tab and edit at least some of the member-visible display content (you must make at least one edit for the challenge to be visible in web/mobile apps)
6. Integration Test
   a. Publish your game challenge
   b. Look for your game challenge in your web and mobile apps
   c. Do a quick test to see if the game loads and you can see the Game JSON you put above

This is just to get started; the next section discusses how you can iteratively develop your game.


Edit Challenge

Display Name	Sample Game
Internal Name	sample_game ☐ Edit Internal Name
Description	Enter Description (250 characters left)
Type	HTML Game
Prize	0 Point ✓ Auto Approve
Trigger Activity	None
Categories	Select categories...
Tags	Enter Tags
Tracking Code	Enter Tracking Code



Edit Challenge

Prize Picking Strategy

- [✓ Probability]
- Every nth response
- First n response per Day
- Random

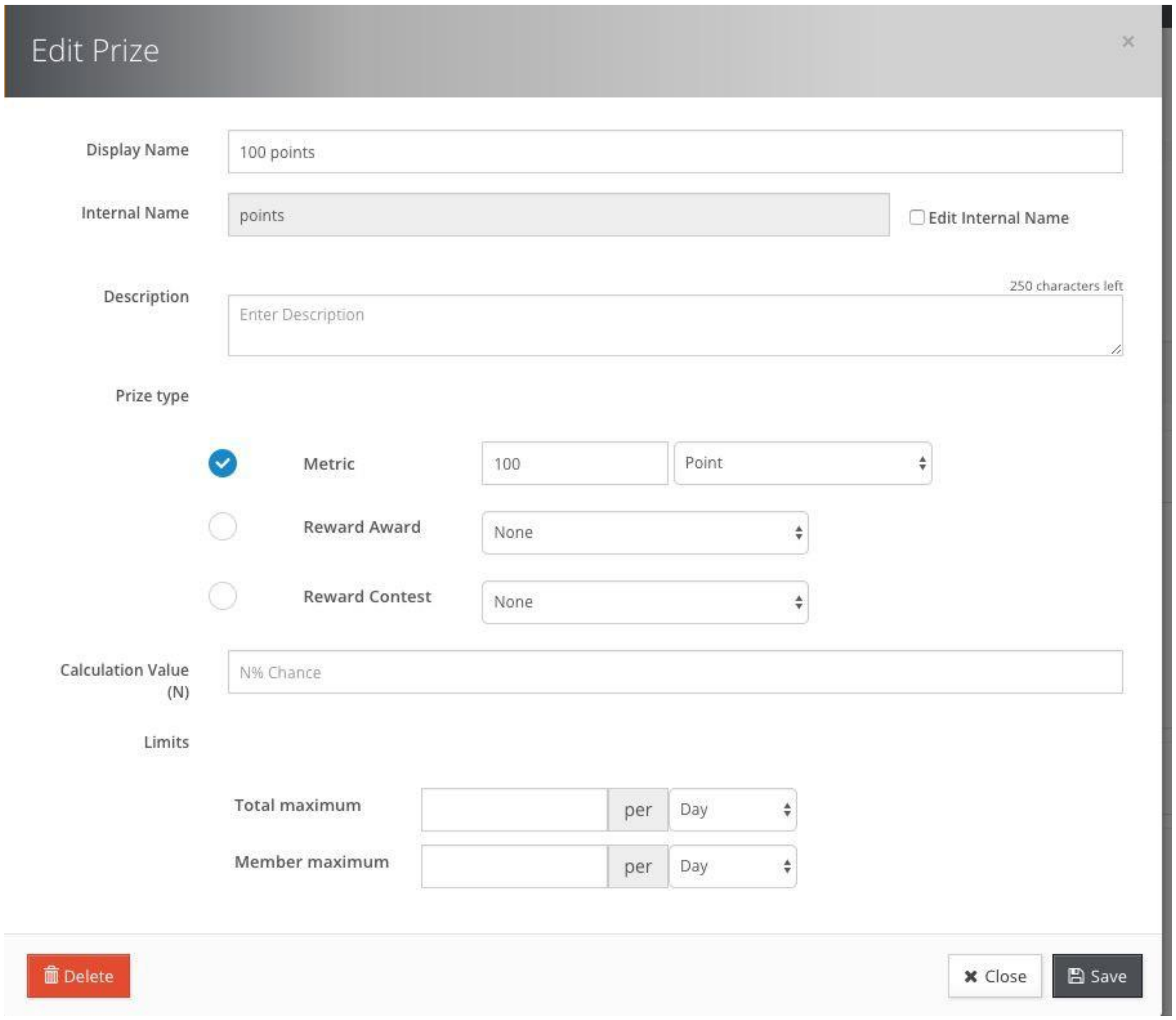
HTML Game

color_number_wheels

JSON Game Options

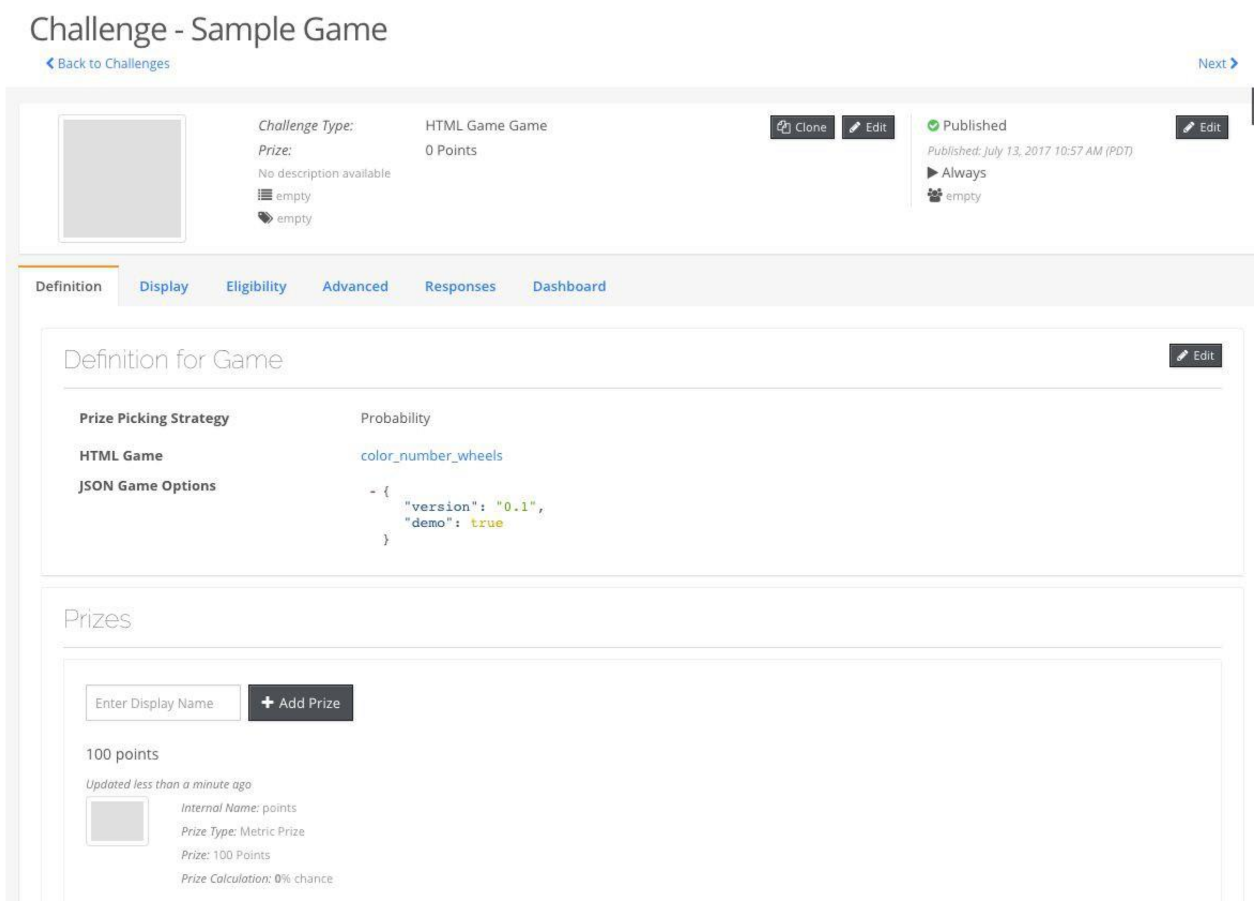
49966 characters left

```
{ "version": "0.1", "demo": true }
```



Edit Prize

Display Name	100 points	
Internal Name	points	☐ Edit Internal Name
Description	Enter Description	250 characters left
Prize type		
✓ Metric	100	Point
○ Reward Award	None	
○ Reward Contest	None	
Calculation Value (N)	N% Chance	
Limits		
Total maximum	per	Day
Member maximum	per	Day


Challenge - Sample Game

‹ Back to Challenges　　Next ›

Challenge Type: HTML Game Game
Prize: 0 Points
No description available
empty
empty

Published
Published: July 13, 2017 10:57 AM (PDT)
▶ Always
empty

Definition for Game

Prize Picking Strategy	Probability
HTML Game	color_number_wheels
JSON Game Options	

```
- {
    "version": "0.1",
    "demo": true
  }
```

Prizes

100 points
Updated less than a minute ago
Internal Name: points
Prize Type: Metric Prize
Prize: 100 Points
Prize Calculation: 0% chance

Step 3: Iterative Development

At this point you should have a basic game content page and game challenge. Getting a game just right takes a lot of persistence and attention to detail. This section provides some tips and specific details.

⚠ Warning: Development should only be done in your Staging Env, not your Production Env. Refer to [Part 1: Game Development Process](), you are probably in Phase 4.

Editing Game Source

If your game content page is hosted by Cheetah Loyalty ("Page Location = Stellar Page"), then you will need to upload file changes to your game. This is straightforward but can be tedious if you need to make a lot of code changes (which is likely. Two dev options we have used in the past are below.

For both options below, it's best to use a publicly accessible web server. While you can use a local or private server, this will prevent other people from playing your game (getting feedback is important), as well as preventing mobile testing.

Minimize index.html

One way to avoid editing hops is to make your index.html page as small as possible, placing all of your javascript and css into an externally loaded file on a web server that you can edit live code from.

```
<!DOCTYPE html> <html lang='en'> <head> <title>Spin</title>
  <style type="text/css"> body { padding: 0px; margin: 0px; background: #fff;
text-align: center; } </style>
  <script src="../phaser.min.js"></script>
  <script src='../stellar-game.js'></script>
  <script src="../imagewheel.js"></script>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head> <body> </body> </html>
```

Temporary External URL

An alternative is to host your page externally ("Page Location = External URL"), but the URL domain must match the web app domain or the browser will refuse to load your file. If you don't have a web server that already matches the domain, you can edit your local **hosts file** to create a fake hostname on the same top-level domain as your web app. The downside is that other users won't be able to view your game, and this won't work for mobile testing.

Definition JSON

Each challenge has a Definition JSON that is passed through to your game at run time, in the onInitializePrizes(prizes, config) function call. You can change these options in the challenge to test various game options.

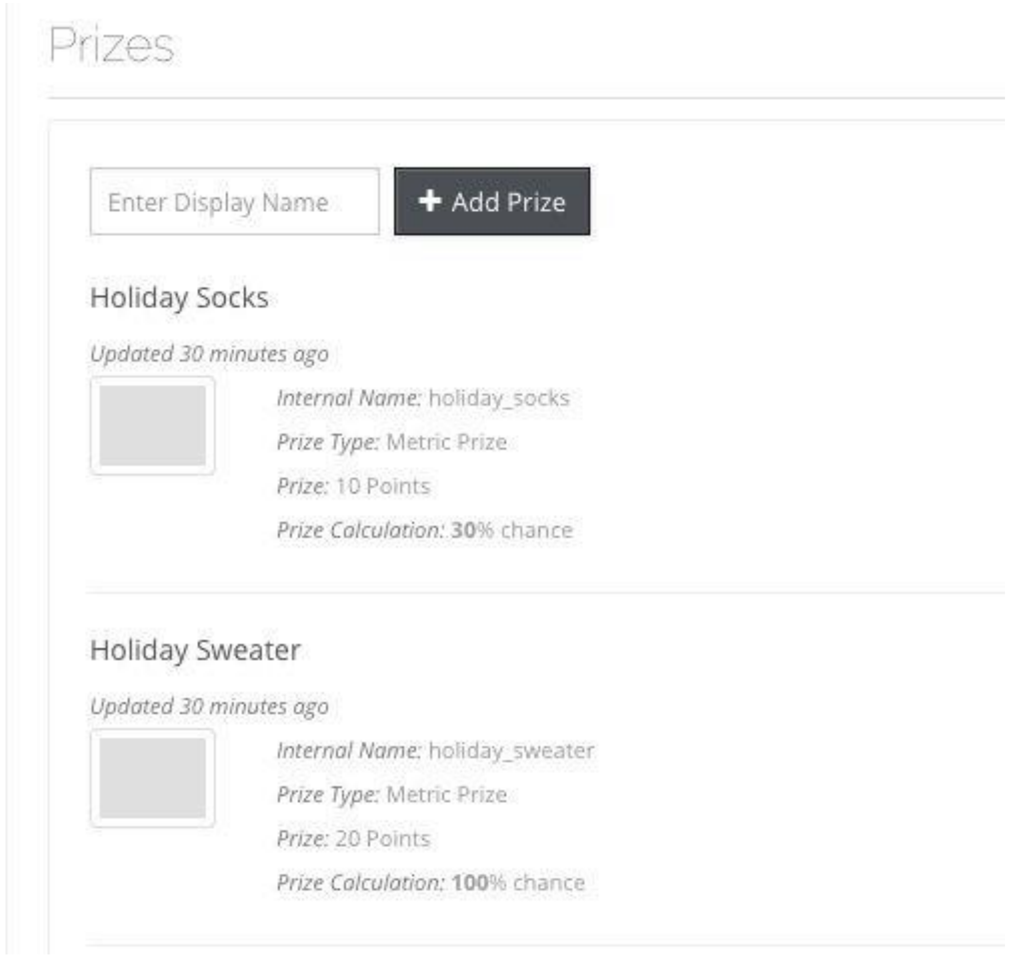
Sometimes it is useful to add a option called "demo: true" in order to test hard-coded demo/test data.

The sample games have a file called "game_config.json". An admin creating a challenge can also copy/paste this into their challenge Definition JSON. If you are using this approach then you should try this copy/paste to make sure the JSON is valid, and so you can see what your marketing admins will see when they try to use your game.. See Appendix A: Game Configuration for more details and tips.

Prizes

Prizes are managed inside the game Challenges. Go to the challenge and then find the Prizes section on the Definition tab, where you can add and edit the prizes of your game challenge.

 Note that this is just a summary of Prizes from a game development standpoint. See the Cheetah Loyalty Product Documentation for full details about Prizes.



Holiday Socks

Updated 30 minutes ago

Internal Name: holiday_socks

Prize Type: Metric Prize

Prize: 10 Points

Prize Calculation: **30**% chance

Holiday Sweater

Updated 30 minutes ago

Internal Name: holiday_sweater

Prize Type: Metric Prize

Prize: 20 Points

Prize Calculation: **100**% chance

Use Realistic Prizes

While developing your game, configure prizes that you expect your marketers to use. For example, the Image Wheel game usually has a few different prizes that award points, with each prize corresponding to a wheel slice.

Getting the list of possible prizes

The prizes are passed through to your game at run time, in the onInitializePrizes(prizes, gameConfig) function call. This is an array of all prizes configured in the challenge, NOT the prize that the user has won.

For example, here is the prizes parameter value received for a spin-the-wheel game:

```
[{"id":6,"internal_name":"orange10","label":"Orange for 10
points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"10 Points"},{"id":7,"internal_name":"blue15","label":"Blue for
15 points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"15 Points"},{"id":9,"internal_name":"green30","label":"Green
for 30 points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"30 Points"},{"id":10,"internal_name":"red20","label":"Red for
20 points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"20 Points"},{"id":11,"internal_name":"black25","label":"Black
for 25 points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"25 Points"},{"id":13,"internal_name":"purple5","label":"Purple
for 5 points","image_url":"","thumb_image_url":"","prize_type":"Metric
Prize","prize":"5 Points"}]
```

Mapping Prizes to Game Outcomes

If your game needs to map prizes to game outcomes, you can use the **internal_name** of each prize. Each prize has an internal_name that is unique within the prizes in that challenge.

For example, the Image Wheel game usually has a few different prizes that award points, with each prize corresponding to a wheel slice. In the sample prizes data above, the prize with **internal_name=red20** corresponds to the portion of the wheel image that says is colored red and shows text “20”. There is a corresponding mapping of the 16 slice names in the game_config.json file, which maps the slice names to the 16 slices by position. Other games might map prizes to game outcomes in different ways, but this worked well for the wheel.

```
...{"id":13,"internal_name":"purple5",...

..."sliceNames": [ "orange10", "black25", "blue15",  "purple5",
                "black25",  "green30", "red20",   "blue15",
                "purple5",  "black25",  "green30", "orange10",
                "blue15",   "purple5", "black25",   "red20" ],...
```

Note: See Appendix A for the full game_config.json from this example.

Prize Determination

Prizes are determined by the Cheetah Loyalty API when your game calls Stellar.game.submitChallenge() method. The prize won is returned to your game in the onFinishPrizeSubmissionHandler() callback, in the prizeId parameter. The **prizeId** corresponds to the integer prize_id in the prizes list data.

```
...{"id":13,"internal_name":"purple5",...
```

Prizes are determined entirely by challenge settings, such as the Prize Picking Strategy setting found in the Challenge Definition, and by Prize settings. Details of these settings are in the Cheetah Loyalty Product Documentation.

Appendix A: Game Configuration

Each game type can be used for many challenges. Making your game flexible and easy to configure helps you to get the most benefit from of your game. The “Image Wheel” game allows the marketers to specify the number of slices, the slice names, and the various images such as background and wheel, making this game easy to reuse, re-skin, and customize over time.

Try to avoid hardcoding any possibly useful configuration options into your game, instead making them configurable. The sample games have three layers of configuration.

1. **config object:** a “config” variable at the top of the javascript game file with default values
2. **game_config.json:** file loaded when the game initializes, overrides config object
3. **gameConfig:** object passed at runtime in onInitializePrizes, which comes from the Challenge Definition JSON

Each layer is overridden by the next. So, the game_config.json only needs to include configuration options you want to make “public”, and the admin configuring the Challenge Definition JSON only needs to include specific options that need to be overridden.

For example, here is the game_config.json for one of the Image Wheel games.

```
{
"game": "imagewheel",
"variant": "colornumbers",
"numSpins": 1,
"slices": 16,
"sliceNames": [ "orange10", "black25", "blue15",  "purple5",
                "black25",  "green30", "red20",   "blue15",
                "purple5",  "black25",  "green30", "orange10",
                "blue15",   "purple5", "black25",   "red20" ],
"aspectRatio": 1.61,
"marginTop": 70,
"wheelSize": 480,
"backgroundColor": "#FFFFFF",
"backgroundEnabled": true,
"pointerEnabled": true,
"centerEnabled": false,
"prizeTextEnabled": false,
"backgroundImage": "background.png",
"wheelImage": "wheel.png",
"centerImage": "center.png",
"pointerImage": "pointer.png"
}
```

See Appendix C: Sample Game Code for source code of the Image Wheel sample game, which uses the above strategy.

Appendix B: stellar-game.js

This is the source code of stellar-game.js for your reference.

The specific implementation is subject to change, although the **interface** will remain stable. Therefore, always use the live hosted version from your program environment. Load this script in your index.html, as shown in the sample games.

```
<script src="<static server hostname>/sdk/games/stellar-game.js"></script>
```

Note that your server hostname may be different in staging versus production, so an option is to upload a copy of stellar-game.js into your Content Page and then reference it with a local path.

```
<script src="stellar-game.js"></script>

/**
 * Stellar Game SDK
 * Use this SDK to show HTML5 games in Stellar web and mobile apps.
 * The web or mobile app is the "game container".
 */

/**
 *  Sample Implementation
 *  You need to implement the following methods and register them with
Stellar.game.
 */
/*
 (function() {
  // This function will be called by the game container.
  Stellar.game.onInitializePrizeHandler = function (prizes, gameConfig) {
    console.log("onInitializePrizeHandler", prizes, gameConfig);
  };

  // Call this function to award the prize
  console.log("calling Stellar.game.submitChallenge");
  Stellar.game.submitChallenge();

  // One of these functions will be called by the game container as a callback
from Stellar.game.submitChallenge.
  Stellar.game.onFinishChallengeSubmissionHandler = function (prizes, prizeId)
{
    console.log("onFinishChallengeSubmissionHandler", prizes, prizeId);
    // Call this function when the game is done and you want to show the prize
    Stellar.game.showResult();
  };
  Stellar.game.onFinishChallengeSubmissionWithErrorHandler = function (error) {
    console.log("onFinishChallengeSubmissionWithErrorHandler", error);
    // The same function that shows the prize will also display an error
message
    Stellar.game.showResult();
  };

  // Call to send custom message to game container
  Stellar.game.postMessage({ 'loaded': 'mygame' });

  if (location.hash === "#demo") {
    var prizes = [];
    Stellar.game.demoStart(prizes);
  }


});
*/

/**
 * Stellar Game Implementation
 * Do not modify any of this code directly.
 */
(function(window, document, undefined) {
  var Stellar = {},
    game;

  game = {
    isMobile: false,
    isDemo: false,
    gameConfig: null,
    fileConfig: null,
    onInitializePrizeHandler: function(prizes, gameConfig) {},
    onFinishChallengeSubmissionHandler: function(prizes, prizeId) {
Stellar.game.showResult() },
    onFinishChallengeSubmissionWithErrorHandler: function(error) {
Stellar.game.showResult() },
    onStageTimeoutHandler: function(stage) { Stellar.game.showResult() },
    initializePrizes: function(prizes, gameConfig) {
      game.beginStage(game.stages.initializePrizes);
      console.log('game.initializePrizes', prizes, gameConfig);
      //where prizes = [{"id" : 1, "internal_name": "prize1", "label" : "Prize
Won!"}];
      game.gameConfig = gameConfig || {};
      game.onInitializePrizeHandler(prizes, game.gameConfig);
    },
    submitChallenge: function() {
      game.endStage(game.stages.initializePrizes);
      game.beginStage(game.stages.submitChallenge);
      if (game.isDemo) {
        if (game.demoPrizeId === -1) {
          game.demoPrizeId =
game.demoPrizes[Math.floor(Math.random()*game.demoPrizes.length)].id;
        }
        game.didSubmitChallenge(game.demoPrizes, game.demoPrizeId);
      }
      else if (game.isMobile) {
        mobile.submitChallenge();
      }
      else {
        game.postMessage({ 'gameAction': 'submitChallenge' });
      }
    },
    didSubmitChallenge: function(prizes, prizeId) {
      game.endStage(game.stages.submitChallenge);
      game.beginStage(game.stages.showResult);
      console.log('game.didSubmitChallenge', prizes, prizeId);
      game.onFinishChallengeSubmissionHandler(prizes, prizeId);
    },
    didSubmitChallengeWithError: function(error) {
      game.endStage(game.stages.submitChallenge);
      game.beginStage(game.stages.showResult);
      console.log('game.didSubmitChallengeWithError', error);
      game.onFinishChallengeSubmissionWithErrorHandler(error);
    },
    showResult: function() {
      game.endStage(game.stages.showResult);
      if (game.isDemo) {
        alert("Thank you for playing!");
      }
      else if (game.isMobile) {
        mobile.showResult();
      }
      else {
        game.postMessage({ 'gameAction': 'showResult' });
      }
    },
    close: function() {
      if (game.isMobile) {
        mobile.close();
      }
      else {
        game.postMessage({ 'gameAction': 'close' });
      }
    },
    // Game Stages
    // Change state to "off" to disable the timeout, or just set the timeout
very large
    stages: {
      "initializePrizes": { name: "initialize prizes", timeLimit: 9000, state:
"watch" },
      "submitChallenge": { name: "submit challenge", timeLimit: 8000, state:
"watch" },
      "showResult": { name: "show result", timeLimit: 60000, state: "watch" }
    },
    beginStage: function (stage) {
      var thisStage = stage;
      if (thisStage.state === "watch") {
        thisStage.state = "active";
        window.setTimeout(function () {
          console.log("checking timeout for stage", thisStage);
          if (thisStage.state === "active") {
            thisStage.state = "timeout";
            console.log("processing timeout for stage", thisStage);
            game.onStageTimeoutHandler(thisStage);
          }
        }, thisStage.timeLimit || 5000);
      }
      console.log("Stellar.game.beginStage", stage);
    },
    endStage: function (stage) { stage.state =
      "ended";
      console.log("Stellar.game.endStage", stage);
    },

    demoPrizes: [
      { id: 1, internal_name: "slice1", label: "A KEY!!!" },
      { id: 2, internal_name: "slice2", label: "50 STARS" },
      { id: 3, internal_name: "slice3", label: "500 STARS" },
      { id: 4, internal_name: "slice4", label: "BAD LUCK!!!" },
      { id: 5, internal_name: "slice5", label: "200 STARS" },
      { id: 6, internal_name: "slice6", label: "100 STARS" },
      { id: 7, internal_name: "slice7", label: "150 STARS" },
      { id: 8, internal_name: "slice8", label: "BAD LUCK!!!" }],
    demoPrizeId: -1,
    demoStart: function (demoPrizes, demoPrizeId) {
      game.isDemo = true;
      if (demoPrizes) { game.demoPrizes
        = demoPrizes; game.demoPrizeId
        = demoPrizeId;
      }
      game.initializePrizes(demoPrizes, {});
    },
    postMessageAction: null,
    postMessageChallenge: null,
    postMessage: function(params) {
      params = params || {};
      if (game.postMessageAction) {
        params.action = game.postMessageAction
      }
      if (game.postMessageChallenge) {
        params.challenge = game.postMessageChallenge
      }
      console.log("game.postMessage", params);
      if (!game.isMobile) {
        window.parent.postMessage(JSON.stringify(params), "*");
      }
      else if (mobile.postMessage) {
        mobile.postMessage(JSON.stringify(params));
      }
      else {
        console.log("skipping postmessage");
      }
    },
    receiveMessage: function(event) {
      var data = JSON.parse(event.data);
      if (data) {
        console.log("game.receiveMessage", "parsed data=", data, "from event=",
event);
        if (data.cbAction) {
          game.postMessageAction = data.cbAction;
        }
        if (data.cbChallenge) {
          game.postMessageChallenge = data.cbChallenge;
        }
        if (data.gameAction === 'initializePrizes') {
          game.initializePrizes(data.prizes, data.gameConfig);
        }
        else if (data.gameAction === 'didSubmitChallenge') {
          game.didSubmitChallenge(data.prizes, data.prizeId);
        }

        /*
        if (data.height) {
          var iframe =
document.getElementsByClassName('stellar-contentvis')[0];
          iframe.style.height = data.height + 'px';
          iframe.style.width = '100%'; // Set mobile responsive
        }
        */
      }
      else {
        console.log("game.receiveMessage", "ERROR - could not parse message
json", "from event=", event);
      }
    },
    getConfig: function(defaults) {
      var cfg = extend({}, defaults);
      cfg = extend(cfg, this.gameConfig);
      console.log("getConfig - returning", cfg);
      return cfg;
    },
    getFileConfig: function(defaults, cb, filePath) {
      filePath = filePath || './game_config.json';
      var cfg = extend({}, defaults);
      if (this.fileConfig) {
        cfg = extend(cfg, this.fileConfig); cfg =
        game.getConfig(cfg);
        console.log("getFileConfig returning", cfg);
        return cfg;
      }
      else if (this.fileConfigState) {
        console.log("getFileConfig skipping - already loading gameConfig",
this.fileConfigState);
      }
      else {
        game.loadFileConfig(function() {
          console.log("getFileConfig - returning", this.fileConfig);
          cfg = extend(cfg, this.fileConfig)
          cfg = game.getConfig(cfg);
          cb(cfg);
        }, filePath);
      }
    },
    loadFileConfig: function(cb, filePath) {
      this.fileConfigState = 'loading';
      console.log("loadFileConfig - starting", filePath);
      game.loadJSON(filePath, function(json) {
        this.fileConfigState = 'loaded';
        console.log("loadFileConfig - SUCCESS", filePath, json);
        if (json) {
          this.fileConfig = json;
        }
        else {
          this.fileConfigState = 'error';
        }
        cb(this.fileConfig);
      },
      function () {
        this.fileConfigState = 'error';
        console.log("loadFileConfig - ERROR", filePath, arguments);
        this.configError = true;
        cb(this.fileConfig);
      });
    },
    loadJSON: function(filePath, success, error) {
      var xhr = new XMLHttpRequest();
      xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
```

```
          if (xhr.status === 200) {
            if (success) { success(JSON.parse(xhr.responseText)) }
          } else {
            if (error) { error(xhr) }
          }
        }
      };
      xhr.open("GET", filePath, true);
      xhr.send();
    }
  };

  Stellar.game = game;

  if (window.addEventListener) {
    window.addEventListener("message", game.receiveMessage, false);
  }
  else if (window.attachEvent) {
    window.attachEvent("onmessage", game.receiveMessage);
  }

  function extend(a, b){
    if (a && typeof a === 'object' && b && typeof b === 'object') {
      for(var key in b)Stellar.game.assets = {};
Stellar.game.boot = function (game) {}
Stellar.game.boot.prototype = {
    preload: function(){
      var self = this;
      Stellar.game.assets.forEach( function (objAsset)
      {
          if(objAsset.type === 'image'){
            self.load.image(objAsset.name, objAsset.path);
          } else if (objAsset.type === 'spritesheet') {
            self.load.spritesheet(objAsset.name, objAsset.path, objAsset.xW,
objAsset.yH, objAsset.frames);
          } else if(objAsset.type === 'audio'){
            self.load.audio(objAsset.name,objAsset.path);
          } else if(objAsset.type === 'video'){
            self.load.video(objAsset.name,objAsset.path);
          } else{
            console.debug('Type undefined : ', objAsset.type );
          }
      });
      this.load.onLoadComplete.add(this.loadComplete, this);

    },
    update:function (){ var that
        = this; if(that.ready
        === true){
          // Stellar.game.postMessage({ 'canHideLoader': true });
```

```
                console.log("game start : hide loader");
                that.state.start('PlayGame');
              }
          },
          loadComplete: function(){
            var that = this;
            that.ready = true;
          }
      };

              if(b.hasOwnProperty(key))
                  a[key] = b[key];
          }
          return a;
      }


      Stellar.game.assets = {};
      Stellar.game.boot = function (game) {}
      Stellar.game.boot.prototype = {
          preload: function(){
            var self = this;
            Stellar.game.assets.forEach( function (objAsset)
            {
                if(objAsset.type === 'image'){
                  self.load.image(objAsset.name, objAsset.path);
                } else if (objAsset.type === 'spritesheet') {
                  self.load.spritesheet(objAsset.name, objAsset.path, objAsset.xW,
      objAsset.yH, objAsset.frames);
                } else if(objAsset.type === 'audio'){
                  self.load.audio(objAsset.name,objAsset.path);
                } else if(objAsset.type === 'video'){
                  self.load.video(objAsset.name,objAsset.path);
                } else{
                  console.debug('Type undefined : ', objAsset.type );
                }
            });
            this.load.onLoadComplete.add(this.loadComplete, this);

          },
          update:function (){ var that
              = this; if(that.ready
              === true){
                // Stellar.game.postMessage({ 'canHideLoader': true });
                console.log("game start : hide loader");
                that.state.start('PlayGame');
              }
          },
          loadComplete: function(){
            var that = this;
```

```
        that.ready = true;
      }
  };

    /*
    var height = document.innerHeight; var
    width = document.innerWidth;
    game.postMessage('loaded=1', "*");
    game.postMessage('height='+height, "*");
    game.postMessage('width='+width, "*");
    */

    window.Stellar = Stellar;

    return Stellar;
  })(window, document);
```

Appendix C: Sample Game Code

This is the source code for one of the sample games, the "Image Wheel". This is a spin-the-wheel game that involves a spinning image.

Note that the actual game in your program sample games may have been updated since this document was written.

```
/*
 * Image Wheel
 */

(function() {

  //
  // Configuration Options
  // These should are overridden by game_config.json
  //
  var config = { "slices":
    16, "sliceNames": null,
    "aspectRatio": 1.61,
    "marginTop": 20,
    "wheelSize": 470,
    "accuracy_degrees": true,
    "accuracyStarterPoint": 0.9,
    "scaleBackgroundEnabled": false,
    "centerEnabled": false,
    "pointerEnabled": true,
    "backgroundColor": "#000000",
    "backgroundEnabled": true,
    "prizeTextEnabled": false,
    "backgroundImage": "background.png",
    "wheelImage": "wheel.png",
    "centerImage": "center.png",
    "autoShowMillis": 1000,
    "pointerImage": "pointer.png",
    "numSpins": 1,
    "debug": false
  };

  // the number of prize slices
  // prize names, starting from 12 o'clock going clockwise
  // these slices are for testing/demos
  var prizeSlices = [];

  //
  // Implementation
  //
  var game;
  var wheel;
  var canSpin;
  // the prize you are about to win
  var prize;
  var degrees;
  // text field where to show the prize
  var prizeText;

  // scaling
  var deviceRatio = (window.innerWidth/window.innerHeight); //device aspect
ratio
  var aspectRatio, dpr, width, height, scaleRatio;

  function getPrize(prizeId) {
    console.log("getPrize", prizeSlices, prizeId);
    prizeId = prizeId || 0; // 0 is no prize
    // find the possible slices that match the prize (could be several)
    var sliceChoices = [];
    for (var i = 0; i < prizeSlices.length; i++) {
      if (''+prizeId === ''+prizeSlices[i].id) {
        sliceChoices.push(prizeSlices[i]);
      }
    }
    if (sliceChoices.length === 0) {
      console.log('error - no prize choices!');
      sliceChoices = prizeSlices;
    }
    // pick a random slice from the choices we found
    prize = sliceChoices[Math.floor(Math.random() * sliceChoices.length)];

    if (prize) {
      // we now have the prize and prizeSlice
      var sliceIndex = prize.sliceIndex;
      degrees = Math.floor((config.slices - sliceIndex - 0.2 - Math.random() *
.5) * 360 / config.slices);
      console.log("degrees=", degrees);
      if (config.pointerEnabled) {
        degrees = degrees + config.slices / 2 * 360 / config.slices;
        if (config.accuracy_degrees === true) {
          degrees = (360 / config.sliceNames.length) * sliceIndex;
        }
        console.log("degrees pointer=", degrees);
      }
      console.log("getPrize", "prizeId=", prizeId, "sliceIndex=", sliceIndex,
"prize=", prizeSlices[sliceIndex], "degrees=", degrees);
    }
    else {
      console.log('error - no prize!');
    }
  }

  function initGame() {
    aspectRatio = config.aspectRatio * deviceRatio;
    dpr = Math.min(window.devicePixelRatio, 2.2);
    width = window.innerWidth;
    height = window.innerHeight;
    // Set aspect ratio. This game is tall so the height is the primary
dimension.
    width = height / aspectRatio;
    if (width > window.innerWidth) {
      width = window.innerWidth;
      height = width * aspectRatio;
    }
    // The mimum scale ratio of .8 applies to web
    scaleRatio = width / config.wheelSize * 0.9; // Math.max(0.8, dpr/3);
    var scaleDebug = "SCALING width:"+width+" height:"+height+" dpr:"+dpr+"
innerWidth:"+window.innerWidth+" innerHeight:"+window.innerHeight+"
aspectRatio:"+aspectRatio;
    console.log(scaleDebug);
    if (config.debug) {
      alert(scaleDebug);
    }
    game = new Phaser.Game("100%", "100%", Phaser.CANVAS, null, null, true);
    //Global.game.scale.scaleMode = Phaser.ScaleManager.RESIZE
    //game.scale.scaleMode = Phaser.ScaleManager.SHOW_ALL;
    // adding "PlayGame" state
    game.state.add("PlayGame", playGame);
    // launching "PlayGame" state
    game.state.start("PlayGame");
    //game.stage.scale.scaleMode = Phaser.ScaleManager.SHOW_ALL;
    //game.stage.scale.setShowAll();
    //window.addEventListener('resize', function () {
game.stage.scale.refresh() });
    //game.stage.scale.refresh();
  }
  // PLAYGAME STATE
  var playGame = function(game) {};

  playGame.prototype = {
    // function to be executed once the state preloads
    preload: function() {
      // preloading graphic assets
      game.load.image("wheel", config.wheelImage);
      game.stage.disableVisibilityChange = true;
      if (config.backgroundEnabled) { game.load.image("background",
config.backgroundImage) }
```

```
      if (config.centerEnabled) { game.load.image("center", config.centerImage)
}
      if (config.pointerEnabled) { game.load.image("pointer",
config.pointerImage) }
    },
    // funtion to be executed when the state is created
    create: function() {
      // giving some color to background
      game.stage.backgroundColor = config.backgroundColor;
      if (config.backgroundEnabled) {
        var background = game.add.sprite(width/2, 0, 'background')
        if (config.scaleBackgroundEnabled === true) {
          background.scale.setTo(scaleRatio*1.125, scaleRatio*1.125);
        }
        background.anchor.set(0.5, 0);
      }
      // adding the wheel in the middle of the canvas
      var wheelSize = config.wheelSize * scaleRatio,
          wheelCenter = wheelSize/2 + config.marginTop*scaleRatio,
          wheelBottom = wheelSize + config.marginTop*scaleRatio;
      wheel = game.add.sprite(width/2, wheelCenter, "wheel");
      wheel.scale.setTo(scaleRatio, scaleRatio);
      wheel.anchor.set(0.5);

      if (config.centerEnabled) {
        var center = game.add.sprite(width/2, wheelCenter, "center");
        center.scale.setTo(scaleRatio, scaleRatio);
        center.anchor.set(0.5);
      }
      if (config.pointerEnabled) {
        var pointer = game.add.sprite(width/2, wheelBottom, "pointer");
        pointer.scale.setTo(scaleRatio, scaleRatio);
        pointer.anchor.set(0.5, 0);
      }
      if (config.prizeTextEnabled) {
        // adding the text field
        var textStyle = {
          font: "32px Verdana",
          fill: "#333333",
          align: "center"
        };
        prizeText = game.add.text(width/2, wheelBottom, "Spin!", textStyle); //
500
        prizeText.scale.setTo(scaleRatio, scaleRatio);
        prizeText.anchor.set(0.5);
      }
      // the game has just started = we can spin the wheel
      canSpin = true;
      // waiting for your input, then calling "spin" function
      game.input.onDown.add(this.spin, this);
    },
    // function to spin the wheel
    spin: function() {
      // can we spin the wheel?
      if (numSpins > 0) {
        // setup prize callback
        var that = this;
        canSpin = false;
        numSpins--;
        var rounds = game.rnd.between(5, 8),
            angle = 360*rounds*config.accuracyStarterPoint, // 0.9 prevents the
wheel from returning to its starting point
            angleMillis = 3;
        var spinTweenA = game.add.tween(wheel).to({ angle: "+"+angle },
angle*angleMillis, Phaser.Easing.Quadratic.In);
        Stellar.game.onFinishChallengeSubmissionHandler = function(prizes,
prizeId) {
          getPrize(prizeId);
          // calculate second tween angle from starting 0 degrees to support
random offset in first tween
          rounds += 2;
          angle = 360*rounds + degrees - angle;
          var spinTweenB = game.add.tween(wheel).to({ angle: '+'+angle },
angle*angleMillis, Phaser.Easing.Quadratic.Out);
          // once the tween is completed, call winPrize function
          spinTweenA.chain(spinTweenB);
          spinTweenB.onComplete.add(that.winPrize, that);
        };
        spinTweenA.start();

        if (config.prizeTextEnabled) {
          // resetting text field
          prizeText.text = "";
        }
        // find the prize
        Stellar.game.submitChallenge();
      }
      else if (prize) {
        window.setTimeout(function () { if (Stellar) {
Stellar.game.showResult() }}, config.autoShowMillis);
      }
    },
    // function to assign the prize
    winPrize: function() {
      console.log("winPrize");
      var that = this;
      // allow user to spin the wheel again?
      canSpin = numSpins > 0;
      if (config.prizeTextEnabled) {
        // writing the prize you just won
        prizeText.text = prize.prizeText || prize.label;
      }
      //game.time.events.add(Phaser.Timer.SECOND * 2, that.showPrize, that);
      that.showPrize();
    },
    showPrize: function() {
      console.log("showPrize");
      window.setTimeout(function () { if (Stellar) { Stellar.game.showResult()
}}, config.autoShowMillis);
    }
  }

  function buildPrizeSlices (prizes) {
    prizeSlices = [];

    var slice;

    for (var i=0; i<config.slices; i++) {
      slice = { id: 0, sliceIndex: i, label: "", prizeText: "Thanks for\n
playing" };

      if (config.sliceNames) {
        slice.internal_name = config.sliceNames[i];
      }
      else {
        slice.internal_name = 'slice'+i;
      }

      prizeSlices.push(slice);

    }

    for (var j = 0; j < prizes.length; j++) {
      prizes[j].sliceCount = 0;
      if (!prizes[j].internal_name) {
        prizes[j].internal_name = prizeSlices[j].internal_name;
      }
      for (var i=0; i<config.slices; i++) {
        slice = prizeSlices[i];
        if (slice.internal_name === prizes[j].internal_name) {
          prizes[j].sliceCount++;
          slice.id = prizes[j].id;
          slice.label = prizes[j].label;
          slice.prizeText = "You win!";
        }
      }
      if (prizes[j].sliceCount === 0) {
        console.log("warning - prize not used by any slice", prizes[j]);
      }
    }
    console.log("buildPrizeSlices", prizeSlices);
  }

  Stellar.game.onInitializePrizeHandler = function(prizes, gameConfig) {
    console.log("onInitializePrizeHandler", prizes, gameConfig);
    if (!prizes || prizes.length === 0) {
      return;
    }

    Stellar.game.postMessage({ 'ack': 'prizes' });
    Stellar.game.onFinishChallengeSubmissionWithErrorHandler = function(error)
{};


    Stellar.game.getFileConfig(config, function (gameConfig) {
      console.log("imagewheel gameConfig", gameConfig);
      config = gameConfig;
      numSpins = config.numSpins;

      Stellar.game.postMessage({
        'loaded': 'imagewheel'
      });
      buildPrizeSlices(prizes);
      initGame();

    });
  };

  if (location.hash === "#demo") {
    Stellar.game.demoStart([
      {"id":2,"internal_name":"first_base","label":"First Base for 10
points","image_url":"","prize_type":"Metric Prize","prize":"10 Points"},
      {"id":1,"internal_name":"second_base","label":"Second Base for 20
points","image_url":"","prize_type":"Metric Prize","prize":"20 Points"},
      {"id":3,"internal_name":"third_base","label":"Third Base for 30
points","image_url":"","prize_type":"Metric Prize","prize":"30 Points"},
      {"id":4,"internal_name":"home_base","label":"Home Base for 5
points","image_url":"","prize_type":"Metric Prize","prize":"5 Points"}], 4);
  }

})();
```